(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년12월12일
(11) 등록번호 10-1472321
(24) 등록일자 2014년12월08일

(51) 국제특허분류(Int. Cl.)
G06F 21/55 (2013.01)
(21) 출원번호 10-2013-0066614
(22) 출원일자 2013년06월11일
심사청구일자 2013년06월11일
(56) 선행기술조사문헌
KR1020120084184 A*
KR1020130031035 A*
한국인터넷진흥원 , 2010.11
KR1020120069162 A
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
김휘강
서울 서초구 바우뫼로 91, 111동 1105호 (양재동, 우성아파트)
금영준
서울 성북구 안암로 145, 정보보호대학원 (안암동5가, 고려대학교)
김지홍
서울 성북구 안암로 145, 정보보호대학원 (안암동5가, 고려대학교)
(74) 대리인
특허법인충현

전체 청구항 수 : 총 11 항 심사관 : 구대성

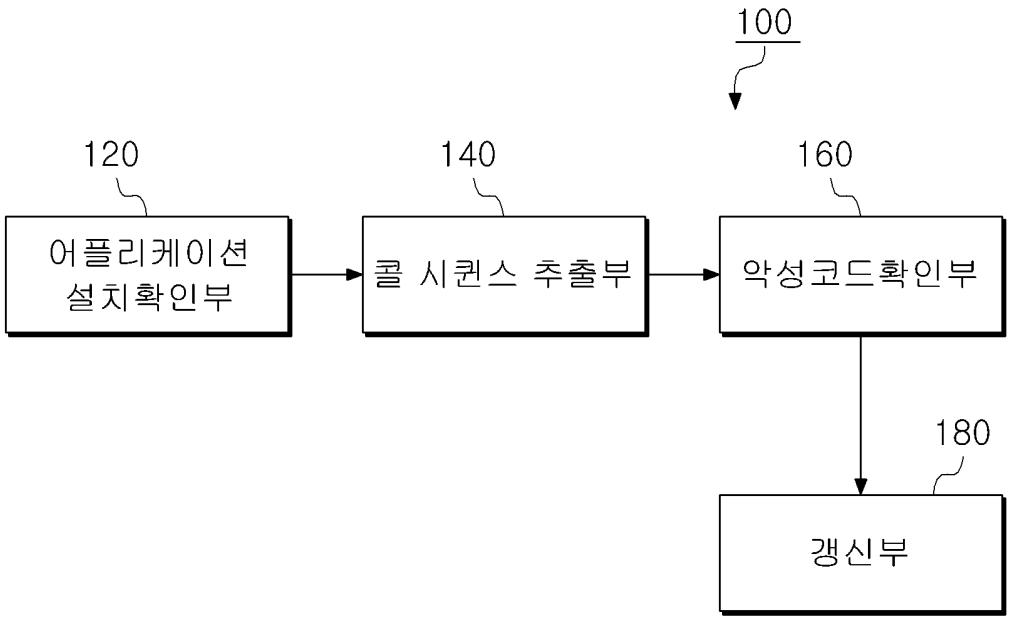
(54) 발명의 명칭 **이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템**

(57) 요 약

본 발명은 이동단말의 어플리케이션 내 악성코드 탐지 시스템 및 방법에 관한 것으로, 보다 구체적으로는 어플리케이션 설치확인부가 이동단말 내 신규 어플리케이션의 설치가 시도되고 있는지 여부를 확인하는 단계; 콜 시퀀스 추출부가 상기 이동단말 내 신규 어플리케이션의 설치가 시도되고 있다고 판단하는 경우, 상기 신규 어플리케

(뒷면에 계속)

대 표 도 - 도1



이션의 설치파일 내 코드로부터 함수에 대한 콜 시퀀스를 추출하는 단계; 및 변종코드 확인부가 추출된 함수의 콜 시퀀스를 데이터베이스 내 기저장된 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 코드로부터 추출된 함수의 콜 시퀀스를 상호 비교하여, 상기 신규 어플리케이션 내 악성코드의 존재유무를 확인하는 단계;를 포함한다.

이러한 구성에 의해, 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템은 이동단말에 신규 어플리케이션의 설치가 시도되는 경우, 그 설치 전에 상기 신규 어플리케이션 내 악성코드가 존재하는지 여부를 미리 확인함으로써, 악성코드가 포함된 어플리케이션이 이동단말에 설치되는 것을 미연에 방지할 수 있는 효과가 있다.

이 발명을 지원한 국가연구개발사업

과제고유번호	H0301-13-1003/1415123340
부처명	미래창조과학부
연구관리전문기관	정보통신산업진흥원
연구사업명	대학 IT연구센터 육성.지원사업
연구과제명	클라우드 환경의 스마트 기기와 서비스 보안 기술 개발 및 연구 인력양성
기 여 율	1/1
주관기관	숭실대학교 산학협력단
연구기간	2013.01.01 ~ 2013.12.31

특허청구의 범위

청구항 1

어플리케이션 설치확인부가 이동단말 내 신규 어플리케이션의 설치가 시도되고 있는지 여부를 확인하는 단계;

콜 시퀀스 추출부가 상기 이동단말 내 신규 어플리케이션의 설치가 시도되고 있다고 판단하는 경우, 상기 신규 어플리케이션의 설치파일 내 코드로부터 함수에 대한 콜 시퀀스를 추출하는 단계; 및

악성코드 확인부가 추출된 함수의 콜 시퀀스를 데이터베이스 내 기저장된 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 코드로부터 추출된 함수의 콜 시퀀스를 상호 비교하여, 상기 신규 어플리케이션 내 악성코드의 존재유무를 확인하는 단계;

를 포함하되,

상기 악성코드의 존재유무를 확인하는 단계는

상기 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스와 상기 데이터베이스 내 기저장된 함수의 콜 시퀀스 간에 유사도를 연산하고, 연산한 유사도값에 기초하여 상기 신규 어플리케이션 내 악성코드의 존재 여부를 판단하는 것을 특징으로 하는 이동단말의 어플리케이션 내 악성코드 탐지 방법.

청구항 2

제1항에 있어서,

상기 추출된 함수의 콜 시퀀스 내 악성코드가 존재한다고 판단한 경우에는 상기 데이터베이스에 앞서 추출된 상기 함수의 콜 시퀀스를 저장하여 업데이트하는 단계;

를 더 포함하는 것을 특징으로 하는 이동단말의 어플리케이션 내 악성코드 탐지 방법.

청구항 3

삭제

청구항 4

제1항에 있어서,

상기 악성코드의 존재유무를 확인하는 단계는

문자열간의 거리 계산(Distance measure) 알고리즘에 기초하여 상기 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스와 상기 데이터베이스 내 기저장된 함수의 콜 시퀀스간에 유사도를 연산하는 것을 특징으로 하는 이동단말의 어플리케이션 내 악성코드 탐지 방법.

청구항 5

제1항에 있어서,

상기 악성코드의 존재유무를 확인하는 단계는

상기 신규 어플리케이션의 설치파일로부터 적어도 하나의 클래스를 추출하고, 추출된 적어도 하나의 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값(Function Call Graph Hash)을 추출하는 과정;

상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값을 상기 데이터베이스로부터 검색하는 과정;

상기 신규 어플리케이션으로부터 추출된 함수 호출 그래프 및 상기 데이터베이스 내 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프를 비교하는 과정;

함수 호출 그래프의 비교 결과에 따라 유사도값을 선택한 후, 각 메소드별 유사도값에 대한 평균값을 연산하여 최종 유사도로 판단하는 과정; 및

상기 최종 유사도가 기설정된 기준 유사도보다 적은 경우에는 상기 어플리케이션 내 악성코드가 존재한다고 판단하는 과정;

을 포함하는 것을 특징으로 하는 이동단말의 어플리케이션 내 악성코드 탐지 방법.

청구항 6

제5항에 있어서,

상기 함수 호출 그래프를 비교하는 과정을 수행하기 전,

상기 신규 어플리케이션으로부터 추출된 함수 호출 그래프 해시값과 상기 데이터베이스에 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프 해시값을 비교하는 과정; 및

함수 호출 그래프 해시값의 비교결과가 동일한 경우에는 상기 신규 어플리케이션 내 악성코드가 존재한다고 판단하는 과정;

을 포함하는 것을 특징으로 하는 이동단말의 어플리케이션 내 악성코드 탐지 방법.

청구항 7

제1항 내지 제2항 및 제4항 내지 제6항 중 어느 한 항에 따른 방법을 컴퓨터로 실행하기 위한 프로그램이 기록된 컴퓨터 판독가능 기록매체.

청구항 8

이동단말 내 신규 어플리케이션의 설치가 시도되고 있는지 여부를 확인하는 어플리케이션 설치확인부;

상기 이동단말 내 신규 어플리케이션의 설치가 시도되고 있다고 판단하는 경우, 상기 신규 어플리케이션의 설치파일 내 코드로부터 함수에 대한 콜 시퀀스를 추출하는 콜 시퀀스 추출부; 및

추출된 함수의 콜 시퀀스를 데이터베이스 내 기저장된 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 코드로부터 추출된 함수의 콜 시퀀스를 상호 비교하여, 상기 신규 어플리케이션 내 악성코드의 존재유무를 확인하는 악성코드확인부;

를 포함하되,

상기 악성코드확인부는

상기 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스와 상기 데이터베이스 내 기저장된 함수의 콜 시퀀스 간에 유사도를 연산하고, 연산한 유사도값에 기초하여 상기 신규 어플리케이션 내 악성코드의 존재 여부를 판단하는 것을 특징으로 하는 이동단말의 어플리케이션 내 악성코드 탐지 시스템.

청구항 9

제8항에 있어서,

상기 추출된 함수의 콜 시퀀스 내 악성코드가 존재한다고 판단한 경우에는 상기 데이터베이스에 앞서 추출된 상기 함수의 콜 시퀀스를 저장하여 업데이트하는 갱신부;

를 더 포함하는 것을 특징으로 하는 이동단말의 어플리케이션 내 악성코드 탐지 시스템.

청구항 10

삭제

청구항 11

제8항에 있어서,

상기 악성코드확인부는

문자열간의 거리 계산(Distance measure) 알고리즘에 기초하여 상기 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스와 상기 데이터베이스 내 기저장된 함수의 콜 시퀀스간에 유사도를 연산하는 것을 특징으로 하는 이동단말의 어플리케이션 내 악성코드 탐지 시스템.

청구항 12

제8항에 있어서,

상기 악성코드확인부는

상기 신규 어플리케이션의 설치파일로부터 적어도 하나의 클래스를 추출하고, 추출된 적어도 하나의 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값(Function Call Graph Hash)을 추출하는 신규FCG추출모듈;

상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 클래스에 속하는 메소드로부터 함수 호출 그래프 및 함수 호출 그래프 해시값을 상기 데이터베이스로부터 검색하는 DB검색모듈;

상기 신규 어플리케이션으로부터 추출된 함수 호출 그래프 및 상기 데이터베이스 내 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프를 비교하는 제1 비교모듈;

함수 호출 그래프의 비교 결과에 따라 유사도값을 선택한 후, 각 메소드별 유사도값에 대한 평균값을 연산하여 최종 유사도로 판단하는 유사도연산모듈; 및

상기 최종 유사도가 기설정된 기준 유사도보다 적은 경우에는 상기 어플리케이션 내 악성코드가 존재한다고 판단하는 악성코드판단모듈;

을 포함하는 것을 특징으로 하는 이동단말의 어플리케이션 내 악성코드 탐지 시스템.

청구항 13

제12항에 있어서,

상기 악성코드확인부는

상기 신규 어플리케이션으로부터 추출된 함수 호출 그래프 해시값과 상기 데이터베이스에 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프 해시값을 비교하는 제2 비교모듈; 을 포함하고,

상기 악성코드판단모듈이

함수 호출 그래프 해시값의 비교결과가 동일한 경우에는 상기 신규 어플리케이션 내 악성코드가 존재한다고 판단하는 것을 더 포함하는 것을 특징으로 하는 이동단말의 어플리케이션 내 악성코드 탐지 시스템.

명 세 서

기 술 분 야

[0001] 본 발명은 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템에 관한 것으로, 특히 이동단말에 설치되는 각종 어플리케이션 중 악성코드가 포함된 어플리케이션이 이동단말에 설치되는 것을 방지할 수 있는 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템에 관한 것이다.

배 경 기 술

[0002] IT 기술이 발전함에 따라 남녀노소를 불문하고, 다양한 계층의 사용자가 스마트폰과 같은 이동단말을 사용하고 있다.

[0003] 이처럼 이동단말이 다양한 계층의 사용자에게 널리 사용될수록, 상기 이동단말은 각종 악성코드에 쉽게 노출되고 있는 것이 현실이다. 이러한 악성코드는 이동단말 내 설치되어, 상기 이동단말의 사용자에 대한 개인정보를 불법적으로 탈취하고, 사용자가 원하지 않는 과금을 부과하며, 상기 이동단말이 다른 이동단말을 공격하는 문제점을 야기한다.

[0004] 특히, 최근에는 일반코드 중 일부가 악성코드와 유사하게 변종되어, 다른 이동단말로 배포됨으로써, 악성코드의 출현이 급격히 증가하는 문제점이 발생하였다.

[0005] 상술한 바와 같이, 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템을 살펴보면 다음과 같다.

[0006] 선행기술 1은 한국공개특허 제2008-0074271호(2008.08.13)로서, 휴대단말 악성코드 처리장치 및 그 처리 방법에 관한 것이다. 이러한 선행기술 1은 휴대단말로 입력되는 정보를 분석하여 상기 휴대단말에 해를 입힐 우려가 있는 악성코드로 의심되는 정보를 검출하는 제1 검출부 및 상기 휴대단말과 유무선망으로 연결된 중앙처리센터로부터 악성코드 검증에 필요한 정보를 수신하여 상기 검출된 악성코드로 의심되는 정보가 실제 악성코드인지 여부를 검증하는 제2 검출부를 포함함으로써, 악성코드 검출 성능을 저하시키지 않으면서 CPU, 메모리 및 통신 오버헤드를 최소화하여 배터리 소모를 대폭 절감할 수 있다.

[0007] 또한, 선행기술 2는 한국공개특허 제2012-0069173호(2012.06.28)로서, 악성코드 자동 분석 시스템 및 그 방법에 관한 것이다. 이러한 선행기술 2는 분석해야할 악성 의심파일들을 FTP(File Transfer Protocol)를 통해 악성코드 분석서버로 업로드(Upload)하는 의심파일 수집서버; 상기 악성코드 분석서버로 특정 파일에 대한 해쉬섬(Hashsum)을 포함하는 악성여부 판단요청정보를 전송하여 악성여부에 대한 판단을 요청하되, 특정 파일 해쉬섬은 XML포맷의 데이터로 HTTP를 이용하여 전송하며, 상기 악성코드 분석서버로부터 특정 파일 해쉬섬에 대응하는 XML포맷 형태의 해쉬섬 악성여부 분석정보를 수신하는 악성코드 탐지서버; 및 사용자로부터 수신한 악성 의심파일 및 사용자 입력정보를 분석하여 악성코드 분석결과정보를 생성하고, 상기 악성 의심파일 수집서버로부터 업로드된 악성 의심파일을 분석하여 악성코드 분석결과정보를 생성하며, 상기 악성코드 탐지서버로부터 수신한 악성여부 판단요청정보에 포함된 특정 파일 해쉬섬의 악성여부를 분석하여 해쉬섬 악성여부 분석정보를 생성하는 악성코드 분석서버;를 포함함으로써, 악성코드 분석서버가 XML포맷 형태의 악성 의심파일을 FTP를 통해 업로드 받아 분석하고, 특정 파일에 대한 해쉬섬을 포함하는 악성여부 판단요청정보를 XML포맷 형태로 수신하여 해당 파일에 대한 악성여부를 분석하여 해쉬섬 악성여부 분석정보를 생성함으로써, 복수개의 악성 의심파일들에 대한 악성코드 여부의 분석 시간을 현저히 저감시키는 효과가 있다.

발명의 내용

해결하려는 과제

[0008] 상기와 같은 종래 기술의 문제점을 해결하기 위해, 본 발명은 이동단말 내 설치하고자 하는 신규 어플리케이션의 코드로부터 콜 시퀀스를 추출하고, 추출한 콜 시퀀스를 기준 콜 시퀀스와 비교하여 유사도를 판단함으로써, 신규 어플리케이션 내 악성코드의 존재여부를 용이하게 파악하는 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템을 제공하고자 한다.

과제의 해결 수단

[0009] 위와 같은 과제를 해결하기 위한 본 발명의 한 실시 예에 따른 이동단말의 어플리케이션 내 악성코드 탐지 방법은 어플리케이션 설치확인부가 이동단말 내 신규 어플리케이션의 설치가 시도되고 있는지 여부를 확인하는 단계; 콜 시퀀스 추출부가 상기 이동단말 내 신규 어플리케이션의 설치가 시도되고 있다고 판단하는 경우, 상기 신규 어플리케이션의 설치파일 내 코드로부터 함수에 대한 콜 시퀀스를 추출하는 단계; 및 변종코드 확인부가 추출된 함수의 콜 시퀀스를 데이터베이스 내 기저장된 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 코드로부터 추출된 함수의 콜 시퀀스를 상호 비교하여, 상기 신규 어플리케이션 내 악성코드의 존재유무를 확인하는 단계; 를 포함한다.

[0010] 보다 바람직하게는 상기 추출된 함수의 콜 시퀀스 내 악성코드가 존재한다고 판단한 경우에는 상기 데이터베이스에 앞서 추출된 상기 함수의 콜 시퀀스를 저장하여 업데이트하는 단계; 를 더 포함한다.

[0011] 특히, 상기 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스와 상기 데이터베이스 내 기저장된 함수의 콜 시퀀스간에 유사도를 연산하고, 연산한 유사도값에 기초하여 상기 신규 어플리케이션 내 악성코드의 존재 여부를 판단하는 상기 악성코드의 존재유무를 확인하는 단계를 포함할 수 있다.

[0012] 특히, 문자열간의 거리 계산(Distance measure) 알고리즘에 기초하여 상기 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스와 상기 데이터베이스 내 기저장된 함수의 콜 시퀀스간에 유사도를 연산하는 상기 악성코드의 존재유무를 확인하는 단계를 포함할 수 있다.

[0013] 보다 바람직하게는 상기 신규 어플리케이션의 설치파일로부터 적어도 하나의 클래스를 추출하고, 추출된 적어도 하나의 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값(Function Call Graph Hash)을 추출하는 과정; 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값(Function Call Graph Hash)을 상기 데이터베이스로부터 검색하는 과정; 상기 신규 어플리케이션으로부터 추출된 함수 호출 그래프 및 상기 데이터베이스 내 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프를 비교하는 과정; 함수 호출 그래프의 비교 결과에 따라 유사도값을 선택한 후, 각 메소드별 유사도값에 대한 평균값을 연산하여 최종 유사도로 판단하는 과정; 및 상기 최종 유사도가 기설정된 기준 유사도보다 적은 경우에는 상기 어플리케이션 내 악성코드가 존재한다고 판단하는 과정; 을 포함하는 상기 악성코드의 존재유무를 확인하는 단계를 포함할 수 있다.

[0014] 보다 바람직하게는 상기 함수 호출 그래프를 비교하는 과정을 수행하기 전, 상기 신규 어플리케이션으로부터 추출된 함수 호출 그래프 해시값과 상기 데이터베이스에 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프 해시값을 비교하는 과정; 및 함수 호출 그래프 해시값의 비교결과가 동일한 경우에는 상기 신규 어플리케이션 내 악성코드가 존재한다고 판단하는 과정;을 포함할 수 있다.

[0015] 위와 같은 과제를 해결하기 위한 본 발명의 다른 실시 예에 따른 이동단말의 어플리케이션 내 악성코드 탐지 시스템은 이동단말 내 신규 어플리케이션의 설치가 시도되고 있는지 여부를 확인하는 어플리케이션 설치확인부; 상기 이동단말 내 신규 어플리케이션의 설치가 시도되고 있다고 판단하는 경우, 상기 신규 어플리케이션의 설치파일 내 코드로부터 함수에 대한 콜 시퀀스를 추출하는 콜 시퀀스 추출부; 및 추출된 함수의 콜 시퀀스를 데이터베이스 내 기저장된 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 코드로부터 추출된 함수의 콜 시퀀스를 상호 비교하여, 상기 신규 어플리케이션 내 악성코드의 존재유무를 확인하는 변종코드 확인부;를 포함할 수 있다.

[0016] 보다 바람직하게는 상기 추출된 함수의 콜 시퀀스 내 악성코드가 존재한다고 판단한 경우에는 상기 데이터베이스에 앞서 추출된 상기 함수의 콜 시퀀스를 저장하여 업데이트하는 갱신부;를 더 포함할 수 있다.

[0017] 특히, 상기 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스와 상기 데이터베이스 내 기저장된 함수의 콜 시퀀스간에 유사도를 연산하고, 연산한 유사도값에 기초하여 상기 신규 어플리케이션 내 악성코드의 존재 여부를 판단하는 변종코드확인부를 포함할 수 있다.

[0018] 특히, 문자열간의 거리 계산(Distance measure) 알고리즘에 기초하여 상기 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스와 상기 데이터베이스 내 기저장된 함수의 콜 시퀀스간에 유사도를 연산하는 변종코드확인부를 포함할 수 있다.

[0019] 보다 바람직하게는 상기 신규 어플리케이션의 설치파일로부터 적어도 하나의 클래스를 추출하고, 추출된 적어도

하나의 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값(Function Call Graph Hash)을 추출하는 신규FCG추출모듈; 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값을 상기 데이터베이스로부터 검색하는 DB검색모듈; 상기 신규 어플리케이션으로부터 추출된 함수 호출 그래프 및 상기 데이터베이스 내 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프를 비교하는 제1 비교모듈; 함수 호출 그래프의 비교 결과에 따라 유사도값을 선택한 후, 각 메소드별 유사도값에 대한 평균값을 연산하여 최종 유사도로 판단하는 유사도연산모듈; 및 상기 최종 유사도가 기설정된 기준 유사도보다 적은 경우에는 상기 어플리케이션 내 악성코드가 존재한다고 판단하는 변종코드판단모듈;을 포함하는 변종코드확인부를 포함할 수 있다.

[0020] 보다 바람직하게는 상기 신규 어플리케이션으로부터 추출된 함수 호출 그래프 해시값과 상기 데이터베이스에 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프 해시값을 비교하는 제2 비교모듈; 을 포함하고, 상기 변종코드판단모듈이 함수 호출 그래프 해시값의 비교결과가 동일한 경우에는 상기 신규 어플리케이션 내 악성코드가 존재한다고 판단하는 것을 더 포함하는 변종코드확인부를 포함할 수 있다.

발명의 효과

[0021] 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템은 이동단말에 신규 어플리케이션의 설치가 시도되는 경우, 그 설치 전에 상기 신규 어플리케이션 내 악성코드가 존재하는지 여부를 미리 확인함으로써, 악성코드가 포함된 어플리케이션이 이동단말에 설치되는 것을 미연에 방지할 수 있는 효과가 있다.

[0022] 또한 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템은 신규 어플리케이션의 실행파일 중 코드의 함수로부터 콜 시퀀스를 추출하고, 상기 신규 어플리케이션과 동일한 어플리케이션에 대해 기저장된 콜 시퀀스를 검색하여 상호 비교한 후 동일여부를 판단함으로써, 상기 신규 어플리케이션이 악성코드가 포함된 어플리케이션인지 여부를 미리 확인할 수 있는 효과가 있다.

[0023] 더불어, 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템은 어플리케이션의 모든 변종에 대하여 시그니처를 작성하지는 않으므로, 이전에 발견되지 않은 악성코드를 용이하게 탐지할 수 있는 효과가 있다.

[0024] 이와 더불어, 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템은 이동단말의 플랫폼 형태로서 전세계적으로 널리 사용되고 있는 안드로이드에서 용이하게 구현되므로, 사용자 편의성을 증대시킬 수 있는 효과가 있다.

도면의 간단한 설명

[0025] 도 1은 본 발명의 일 실시 예에 따른 이동단말의 어플리케이션 내 악성코드 탐지 시스템의 블록도이다.

도 2는 악성코드확인부의 세부구성을 나타낸 블록도이다.

도 3은 본 발명의 다른 실시 예에 따른 이동단말의 어플리케이션 내 악성코드 탐지 방법의 순서도이다.

도 4는 신규 어플리케이션의 실행 파일 내 클래스를 추출하는 과정을 나타낸 도면이다.

도 5는 각 클래스의 메소드로부터 함수 호출 그래프를 추출하는 과정을 나타낸 도면이다.

도 6은 추출한 함수 호출 그래프를 추상화하는 과정을 나타낸 도면이다.

도 7은 신규 어플리케이션 내 악성코드의 존재유무를 확인하는 세부 과정을 나타낸 순서도이다.

도 8은 기저장된 실행파일의 메소드로부터 추출된 FCG를 비교하는 과정을 나타낸 도면이다.

발명을 실시하기 위한 구체적인 내용

[0026] 이하, 본 발명을 바람직한 실시 예와 첨부한 도면을 참고로 하여 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 상세히 설명한다. 그러나 본 발명은 여러 가지 상이한 형태로 구현될

수 있으며, 여기에서 설명하는 실시 예에 한정되는 것은 아니다.

[0027] 이하, 도 1을 참조하여, 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 시스템에 대하여 자세히 살펴보도록 한다.

[0028] 도 1은 본 발명의 일 실시 예에 따른 이동단말의 어플리케이션 내 악성코드 탐지 시스템의 블록도이다.

[0029] 도 1에 도시된 바와 같이, 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 시스템(100)은 어플리케이션 설치확인부(120), 콜 시퀀스 추출부(140), 악성코드확인부(160) 및 갱신부(180)를 포함한다.

[0030] 어플리케이션 설치확인부(120)는 이동단말 내 신규 어플리케이션의 설치가 시도되고 있는지 여부를 확인한다.

[0031] 콜 시퀀스 추출부(140)는 이동단말 내 신규 어플리케이션의 설치가 시도되고 있다고 판단하는 경우에, 상기 신규 어플리케이션의 설치파일 내 코드로부터 함수에 대한 콜 시퀀스를 추출한다. 이때, 추출되는 함수의 콜 시퀀스란, 프로그램의 실행순서를 추상적으로 변경하는 방법으로서, 일부의 변화를 용이하게 탐지할 수 있다.

[0032] 악성코드확인부(160)는 추출된 함수의 콜 시퀀스를 데이터베이스 내 기저장된 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 코드로부터 추출된 함수의 콜 시퀀스를 상호 비교하여 차이여부에 따라, 상기 신규 어플리케이션 내 악성코드가 존재하는지를 판단한다. 즉, 이러한 악성코드확인부(160)는 상기 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스와 상기 데이터베이스 내 기저장된 함수의 콜 시퀀스간에 유사도를 연산하고, 연산한 유사도값에 기초하여 상기 신규 어플리케이션 내 악성코드의 존재 여부를 판단한다. 이때, 사용되는 유사도 연산과정은 문자열간의 거리 계산(Distance measure) 알고리즘에 기초하여 연산될 수 있는데, 이러한 문자열간의 거리 계산 알고리즘 중 하나인 레빈쉬타인 거리 알고리즘은 두 개의 문자열을 상호 비교하여 서로 동일하게 되기까지 몇 개의 글자를 변경해야 하는지를 기준으로 하여 상기 두 개의 문자열 간에 유사도를 연산하는 알고리즘을 나타낸다.

[0033] 도 2는 악성코드확인부의 세부구성을 나타낸 블록도이다.

[0034] 도 2에 도시된 바와 같이, 상기 악성코드확인부(160)는 신규FCG추출모듈(161), DB검색모듈(162), 제1 비교모듈(163), 유사도연산모듈(164), 악성코드판단모듈(165) 및 제2 비교모듈(166)을 포함한다.

[0035] 신규FCG추출모듈(161)은 이동단말 내 설치하고자 하는 신규 어플리케이션의 설치파일로부터 적어도 하나의 클래스를 추출하고, 추출된 적어도 하나의 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값(Function Call Graph Hash)을 추출한다.

[0036] DB검색모듈(162)은 데이터베이스 내 기저장된 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값(Function Call Graph Hash)을 검색한다.

[0037] 제1 비교모듈(163)은 상기 신규 어플리케이션으로부터 추출된 함수 호출 그래프 및 상기 데이터베이스 내 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프를 상호 비교한다.

[0038] 유사도연산모듈(164)은 상기 제1 비교모듈(163)에서 수행한 함수 호출 그래프의 비교 결과에 따라 가장 높은 유사도값을 선택한 후, 각 메소드별 유사도값에 대한 평균값을 연산하여 최종 유사도로 판단한다.

[0039] 악성코드판단모듈(165)은 상기 최종 유사도가 기설정된 기준 유사도보다 적은 경우에는 상기 어플리케이션 내 악성코드가 존재한다고 판단하거나, 함수 호출 그래프 해시값의 비교결과가 동일한 경우에는 상기 신규 어플리케이션 내 악성코드가 존재한다고 판단한다.

[0040] 제2 비교모듈(166)은 신규 어플리케이션으로부터 추출된 함수 호출 그래프 해시값과 데이터베이스에 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프 해시값을 비교한다.

[0041] 다시 도 1로 돌아가서, 갱신부(180)는 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스 내 악성코드가 존재한다고 판단한 경우에는 데이터베이스에 앞서 추출된 상기 함수의 콜 시퀀스를 저장하여 업데이트한다.

[0042] 이하, 도 3을 참조하여 본 발명의 다른 예에 따른 이동단말의 어플리케이션 내 악성코드 탐지 방법을 살펴보도록 한다.

[0043] 도 3은 본 발명의 다른 실시 예에 따른 이동단말의 어플리케이션 내 악성코드 탐지 방법의 순서도이다.

[0044] 도 3에 도시된 바와 같이, 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 방법은 먼저, 어플리케이션 설

치확인부가 이동단말 내 신규 어플리케이션의 설치가 시도되고 있는지 여부를 확인한다(S210).

[0045] 이에 따라, 콜 시퀀스 추출부가 상기 이동단말 내 신규 어플리케이션의 설치가 시도되고 있다고 판단하는 경우, 상기 신규 어플리케이션의 설치파일 내 코드로부터 함수에 대한 콜 시퀀스를 추출한다(S220). 즉, 상기 콜 시퀀스 추출부가 신규 어플리케이션의 설치파일로부터 적어도 하나의 클래스를 추출하고, 추출된 적어도 하나의 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값(Function Call Graph Hash)을 추출한다.

[0046] 이하, 도 4를 참조하여 신규 어플리케이션의 실행파일 내 클래스 추출과정에 대하여 자세히 살펴보면 다음과 같다.

[0047] 도 4는 신규 어플리케이션의 실행 파일 내 클래스를 추출하는 과정을 나타낸 도면이다.

[0048] 도 4(a)에 도시된 바와 같이, 이동단말 내 설치하고자 하는 신규 어플리케이션의 실행파일인 APK파일을 압축해제하여 classes.dex 를 추출한다.

[0049] 이후, 도 4(b)에 도시된 바와 같이, 추출한 파일 classes.dex 를 디셈블(disassemble)하여 생성된 다수의 파일 중 kakao$1.smali 파일을 추출하고, 추출된 kakao$1.smali 파일로부터 도 4(c)에 도시된 바와 같이, 추출된 클래스에 속하는 메소드를 추출한다.

[0050] 도 5는 각 클래스의 메소드로부터 함수 호출 그래프를 호출하는 과정을 나타내는 도면으로서, 도 5에 도시된 바와 같이, send-BB 함수의 함수 호출 그래프를 추출한다.

[0051] 이후, 도 6에 도시된 바와 같이, 앞서 도 5를 통해 추출한 함수 호출 그래프로부터 함수명을 제거하고, 오프셋을 기준으로하여 함수 호출 그래프를 추상화한다.

[0052] 이후, 변종코드 확인부가 앞서 상기 신규 어플리케이션으로부터 추출된 함수의 콜 시퀀스를 데이터베이스 내 기저장된 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 코드로부터 추출된 함수의 콜 시퀀스를 상호 비교하고, 그 비교결과에 따라 상기 신규 어플리케이션 내 악성코드의 존재유무를 확인한다(S230).

[0053] 이하, 도 7을 참조하여 신규 어플리케이션 내 악성코드의 존재유무를 확인하는 과정에 대하여 보다 자세히 살펴보도록 한다.

[0054] 도 7은 신규 어플리케이션 내 악성코드의 존재유무를 확인하는 세부 과정을 나타낸 순서도이다.

[0055] 도 7에 도시된 바와 같이, 신규 어플리케이션 내 악성코드의 존재유무를 확인하는 단계는 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 클래스에 속하는 메소드로부터 함수 호출 그래프(Function Call Graph) 및 함수 호출 그래프 해시값을 미리 저장된 데이터베이스로부터 검색한다(S231).

[0056] 이후, 함수 호출 그래프를 비교하기에 앞서, 상기 신규 어플리케이션으로부터 추출된 함수 호출 그래프 해시값과, 상기 데이터베이스에 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프 해시값을 먼저 비교한다(S232).

[0057] 함수 호출 그래프 해시값의 비교결과가 동일한 경우(S233)에는 유사도값이 1이 되어, 상기 신규 어플리케이션 내 악성코드가 존재한다고 판단한다(S234).

[0058] 또는 함수 호출 그래프 해시값의 비교결과가 동일하지 않는 경우(S233)에는 상기 신규 어플리케이션으로부터 추출된 다수의 함수 호출 그래프(FCG1 ~ FCGn)와, 상기 데이터베이스 내 기저장된 어플리케이션으로부터 추출된 함수 호출 그래프를 비교한다(S235). 즉, 도 8은 도시된 바와 같이, 데이터베이스에 기저장된 함수 호출 그래프(a)와 앞서 추출한 함수 호출 그래프(b)를 상호 비교하여 비교결과에 따라 유사도값을 연산한다. 이때, 유사도값을 연산하기 위해, 문자열간의 거리 계산(Distance measure) 알고리즘을 이용할 수 있다.

[0059] 이어서, 각 함수 호출 그래프별 비교 결과에 따라 가장 높은 유사도값을 선택한 후, 각 메소드별 유사도값에 대한 평균값을 연산하여 연산된 평균값을 최종 유사도로 판단한다(S236). 이때, 유사도값은 0 부터 1 사이에서 획득할 수 있으며, 0 에 가까울수록 동일한 어플리케이션으로 판단할 수 있으며, 1 에 가까운 경우에는 서로 다른 어플리케이션으로 판단하며, 이러한 경우에는 이동단말 내 설치하고자 하는 어플리케이션이 악성코드가 포함된 어플리케이션이라고 판단한다.

[0060] 상기 최종 유사도가 기설정된 기준 유사도보다 적은 경우에는 상기 어플리케이션 내 악성코드가 존재한다고 판단한다(S237).

[0061] 이에 따라, 상기 신규 어플리케이션 내 악성코드가 존재하는지 여부를 판단(S240)하여, 신규 어플리케이션 내 악성코드가 존재한다고 판단한 경우에는 데이터베이스에 앞서 추출한 함수의 콜 시퀀스를 저장하여 업데이트한다(S250).

[0062] 이어서, 상기 신규 어플리케이션 내 악성코드가 포함되어 있음을 이동단말의 사용자에게 전달한다.

[0063] 하지만 이와 달리, 신규 어플리케이션 내 악성코드가 존재하지 않는다고 판단한 경우에는 이동단말 내 상기 신규 어플리케이션을 설치하도록 한다(S260).

[0064] 또한, 이러한 이동단말의 어플리케이션 내 악성코드 탐지 시스템 및 방법은 컴퓨터로 실행하기 위한 프로그램이 기록된 컴퓨터 판독가능 기록매체에 저장될 수 있다. 이때, 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록 장치의 예로는 ROM, RAM, CD-ROM, DVD±ROM, DVD-RAM, 자기 테이프, 플로피 디스크, 하드 디스크(hard disk), 광 데이터 저장장치 등이 있다. 또한, 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 장치에 분산되어 분산방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다.

[0065] 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템은 이동단말에 신규 어플리케이션의 설치가 시도되는 경우, 그 설치 전에 상기 신규 어플리케이션 내 악성코드가 존재하는지 여부를 미리 확인함으로써, 악성코드가 포함된 어플리케이션이 이동단말에 설치되는 것을 미연에 방지할 수 있는 효과가 있다.

[0066] 또한 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템은 신규 어플리케이션의 실행파일 중 코드의 함수로부터 콜 시퀀스를 추출하고, 상기 신규 어플리케이션과 동일한 어플리케이션에 대해 기저장된 콜 시퀀스를 검색하여 상호 비교한 후 동일여부를 판단함으로써, 상기 신규 어플리케이션이 악성코드가 포함된 어플리케이션인지 여부를 미리 확인할 수 있는 효과가 있다.

[0067] 더불어, 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템은 어플리케이션의 모든 변종에 대하여 시그니처를 작성하지는 않으므로, 이전에 발견되지 않은 악성코드를 용이하게 탐지할 수 있는 효과가 있다.

[0068] 이와 더불어, 본 발명의 이동단말의 어플리케이션 내 악성코드 탐지 방법 및 시스템은 이동단말의 플랫폼 형태로서 전세계적으로 널리 사용되고 있는 안드로이드에서 용이하게 구현되므로, 사용자 편의성을 증대시킬 수 있는 효과가 있다.

[0069] 상기에서는 본 발명의 바람직한 실시 예에 대하여 설명하였지만, 본 발명은 이에 한정되는 것이 아니고 본 발명의 기술 사상 범위 내에서 여러 가지로 변형하여 실시하는 것이 가능하고 이 또한 첨부된 특허청구범위에 속하는 것은 당연하다.

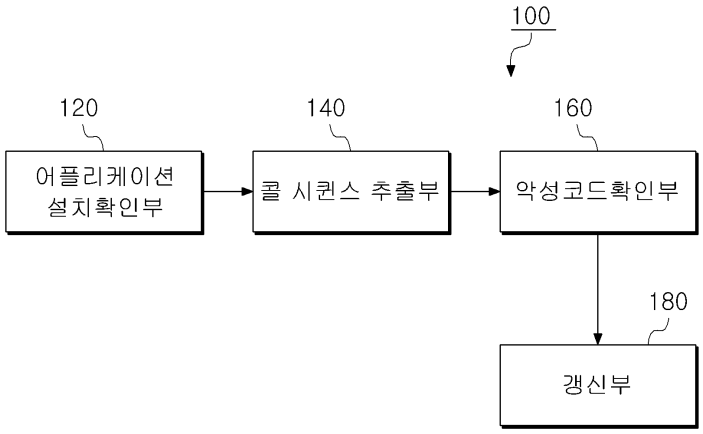
부호의 설명

[0070] 120: 어플리케이션 설치확인부 140: 콜 시퀀스 추출분

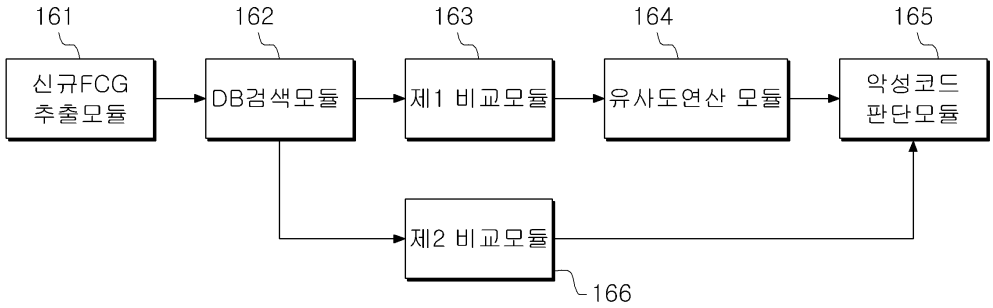
160: 악성코드확인부 180: 갱신부

도면

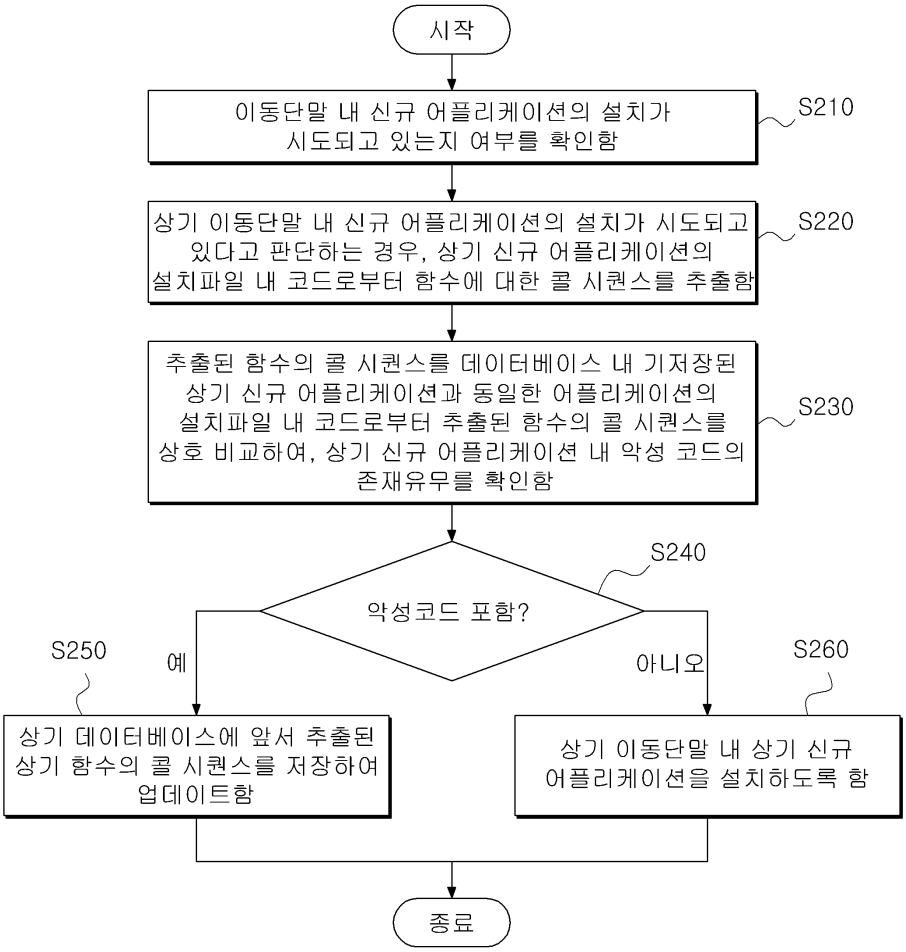
도면1


100
120
어플리케이션 설치확인부
140
콜 시퀀스 추출부
160
악성코드확인부
180
갱신부

도면2


161
신규FCG 추출모듈
162
DB검색모듈
163
제1 비교모듈
164
유사도연산 모듈
165
악성코드 판단모듈
제2 비교모듈
166

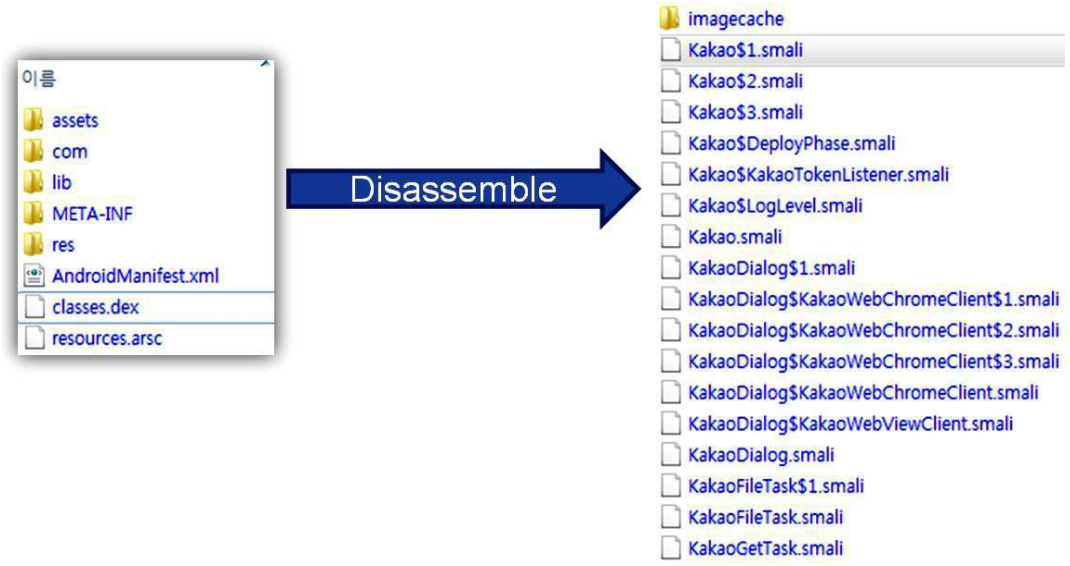
도면3


시작
이동단말 내 신규 어플리케이션의 설치가 시도되고 있는지 여부를 확인함
S210
상기 이동단말 내 신규 어플리케이션의 설치가 시도되고 있다고 판단하는 경우, 상기 신규 어플리케이션의 설치파일 내 코드로부터 함수에 대한 콜 시퀀스를 추출함
S220
추출된 함수의 콜 시퀀스를 데이터베이스 내 기저장된 상기 신규 어플리케이션과 동일한 어플리케이션의 설치파일 내 코드로부터 추출된 함수의 콜 시퀀스를 상호 비교하여, 상기 신규 어플리케이션 내 악성 코드의 존재유무를 확인함
S230
S240
악성코드 포함?
예
아니오
S250
S260
상기 데이터베이스에 앞서 추출된 상기 함수의 콜 시퀀스를 저장하여 업데이트함
상기 이동단말 내 상기 신규 어플리케이션을 설치하도록 함
종료

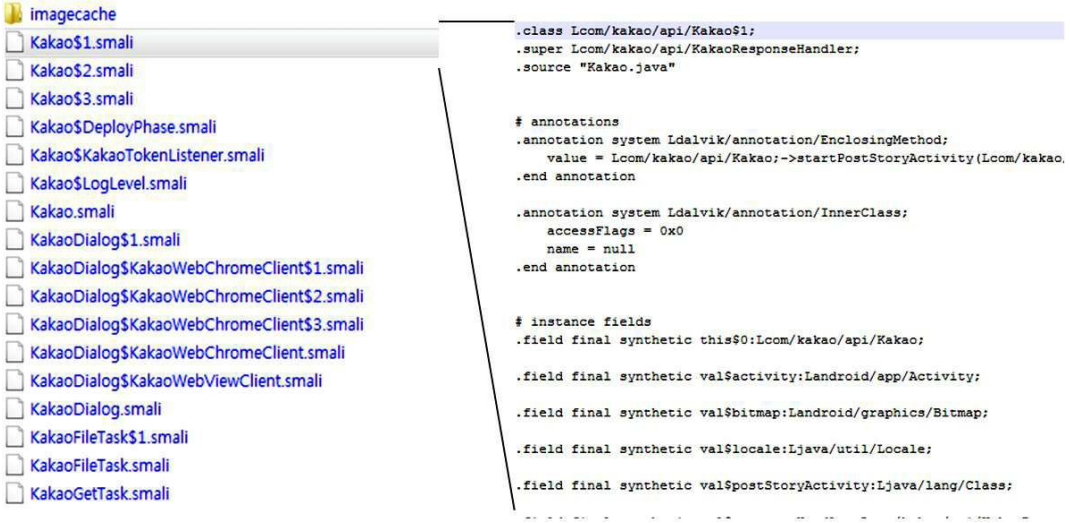
도면4a


APK File
UnZIP
이름
assets
com
lib
META-INF
res
AndroidManifest.xml
classes.dex
resources.arsc

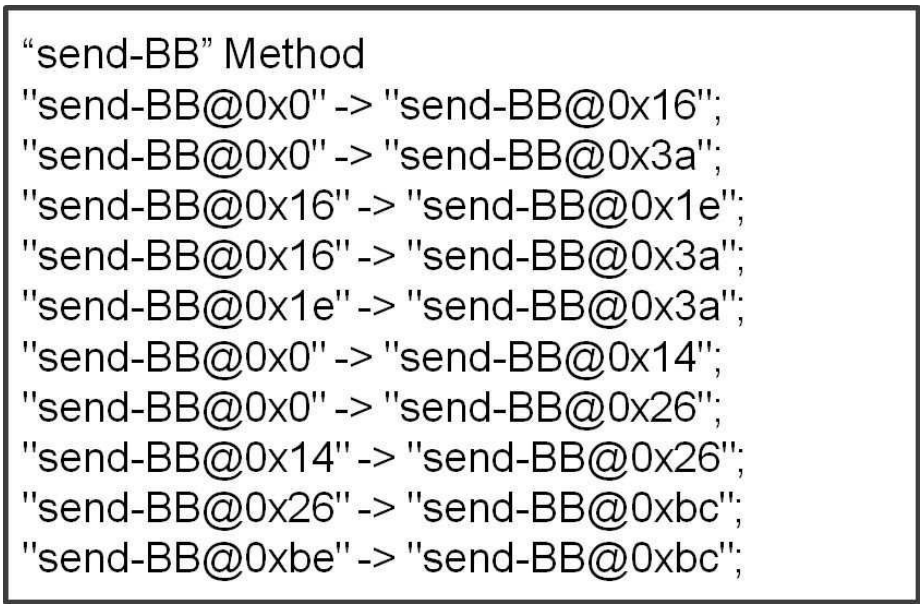
도면4b


이름
assets
com
lib
META-INF
res
AndroidManifest.xml
classes.dex
resources.arsc
Disassemble
imagecache
Kakao$1.smali
Kakao$2.smali
Kakao$3.smali
Kakao$DeployPhase.smali
Kakao$KakaoTokenListener.smali
Kakao$LogLevel.smali
Kakao.smali
KakaoDialog$1.smali
KakaoDialog$KakaoWebChromeClient$1.smali
KakaoDialog$KakaoWebChromeClient$2.smali
KakaoDialog$KakaoWebChromeClient$3.smali
KakaoDialog$KakaoWebChromeClient.smali
KakaoDialog$KakaoWebViewClient.smali
KakaoDialog.smali
KakaoFileTask$1.smali
KakaoFileTask.smali
KakaoGetTask.smali

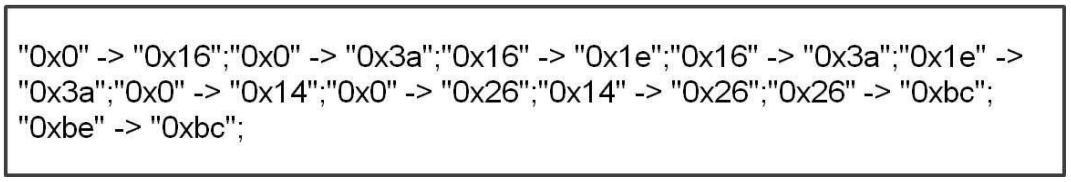
도면4c


imagecache
Kakao$1.smali
Kakao$2.smali
Kakao$3.smali
Kakao$DeployPhase.smali
Kakao$KakaoTokenListener.smali
Kakao$LogLevel.smali
Kakao.smali
KakaoDialog$1.smali
KakaoDialog$KakaoWebChromeClient$1.smali
KakaoDialog$KakaoWebChromeClient$2.smali
KakaoDialog$KakaoWebChromeClient$3.smali
KakaoDialog$KakaoWebChromeClient.smali
KakaoDialog$KakaoWebViewClient.smali
KakaoDialog.smali
KakaoFileTask$1.smali
KakaoFileTask.smali
KakaoGetTask.smali
.class Lcom/kakao/api/Kakao$1;
.super Lcom/kakao/api/KakaoResponseHandler;
.source "Kakao.java"
annotations
.annotation system Ldalvik/annotation/EnclosingMethod;
value = Lcom/kakao/api/Kakao;->startPostStoryActivity(Lcom/kakao,
.end annotation
.annotation system Ldalvik/annotation/InnerClass;
accessFlags = 0x0
name = null
.end annotation
instance fields
.field final synthetic this$0:Lcom/kakao/api/Kakao;
.field final synthetic val$activity:Landroid/app/Activity;
.field final synthetic val$bitmap:Landroid/graphics/Bitmap;
.field final synthetic val$locale:Ljava/util/Locale;
.field final synthetic val$postStoryActivity:Ljava/lang/Class;

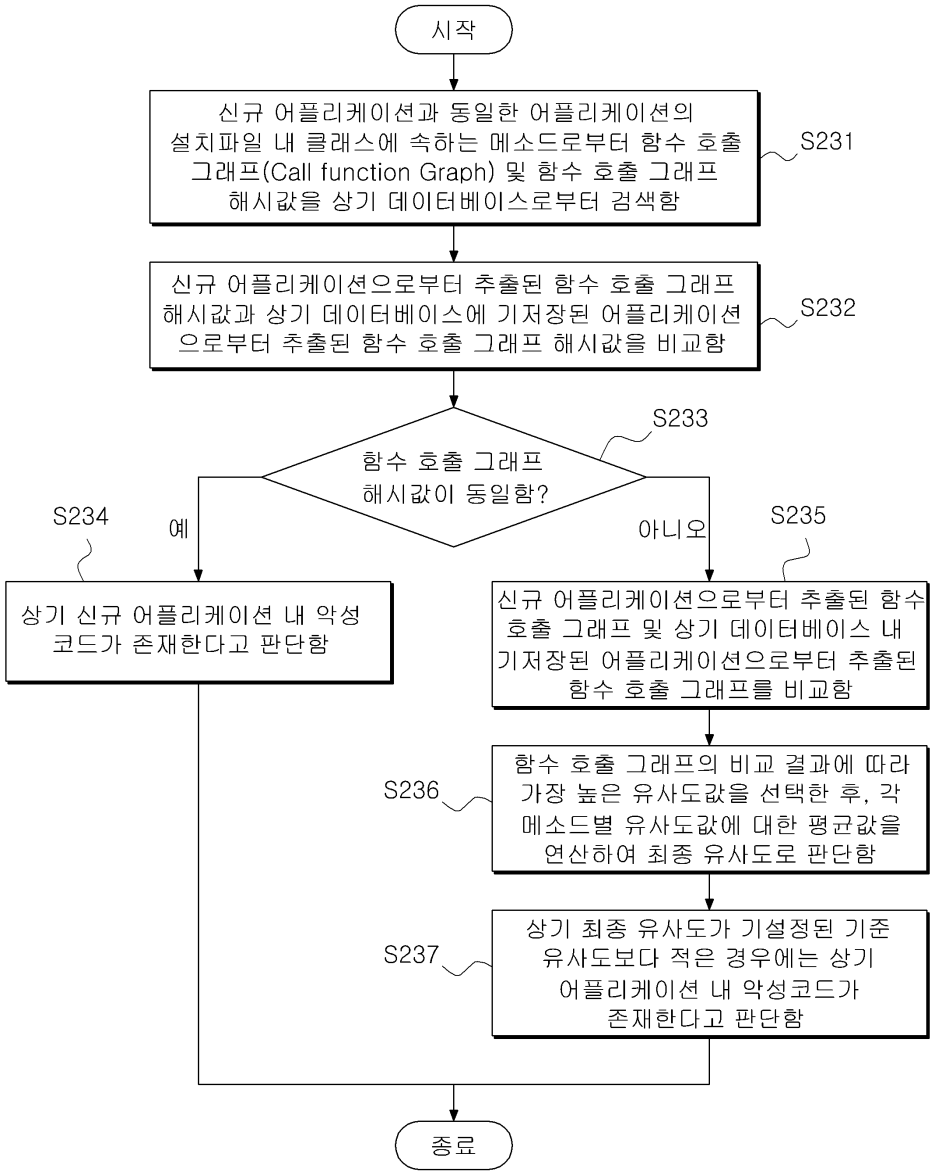
도면5


“send-BB” Method
"send-BB@0x0" -> "send-BB@0x16";
"send-BB@0x0" -> "send-BB@0x3a";
"send-BB@0x16" -> "send-BB@0x1e";
"send-BB@0x16" -> "send-BB@0x3a";
"send-BB@0x1e" -> "send-BB@0x3a";
"send-BB@0x0" -> "send-BB@0x14";
"send-BB@0x0" -> "send-BB@0x26";
"send-BB@0x14" -> "send-BB@0x26";
"send-BB@0x26" -> "send-BB@0xbc";
"send-BB@0xbe" -> "send-BB@0xbc";

도면6


"0x0" -> "0x16";"0x0" -> "0x3a";"0x16" -> "0x1e";"0x16" -> "0x3a";"0x1e" ->
"0x3a";"0x0" -> "0x14";"0x0" -> "0x26";"0x14" -> "0x26";"0x26" -> "0xbc";
"0xbe" -> "0xbc";

도면7



도면8

(a) "0x0" -> "0x16";"0x0" -> "0x3a";"0x16" -> "0x1e";"0x16" -> "0x3a";"0x1e" -> "0x3a";

(b) "0x0" -> "0x16";"0x0" -> "0x3a";"0x16" -> "0x1e";"0x16" -> "0x3a";"0x1e" -> "0x3a";"0x0" -> "0x14";